Part3 Problems 15 to 21

SPPU DBMS PRACTICALS - SET II
Part 3: Problem Statements 15 - 21
==========

Problem Statement 8 — Joins & Subqueries using MySQL

Schema: Employee(Employee_id, First_name, Last_name, hire_date, salary, Job_title, manager_id, department_id)
Departments(Department_id, Department_name, Manager_id, Location_id)
Locations(location_id, street_address, postal_code, city, state, country_id)
Manager(Manager_id, Manager_name)

Queries:
1. SELECT e.First_name, e.Last_name, e.salary FROM Employee e WHERE e.salary > (SELECT salary FROM Employee WHERE last_name = 'Singh' LIMIT 1);
2. SELECT e.First_name, e.Last_name FROM Employee e JOIN Departments d ON e.department_id = d.Department_id JOIN Locations l ON d.Location_id = l.Location_id WHERE e.manager_id IS NOT NULL AND l.country_id = 'US';
3. SELECT First_name, Last_name, salary FROM Employee WHERE salary > (SELECT AVG(salary) FROM Employee);
4. SELECT e.Employee_id, e.Last_name AS Emp_Last, e.manager_id, m.Last_name AS Manager_Last FROM Employee e LEFT JOIN Employee m ON e.manager_id = m.Employee_id;
5. SELECT e.First_name, e.Last_name, e.hire_date FROM Employee e WHERE e.hire_date > (SELECT hire_date FROM Employee WHERE last_name = 'Jones' LIMIT 1);

Problem Statement 9 — Cursors (PL/SQL)

Schema: Employee(Emp_id, Emp_Name, Salary)

1. Explicit cursor:
```
DECLARE
CURSOR c_emp IS SELECT Emp_id, Emp_Name, Salary FROM Employee WHERE Salary > 50000;
v_id Employee.Emp_id%TYPE;
v_name Employee.Emp_Name%TYPE;
v_sal Employee.Salary%TYPE;
BEGIN
OPEN c_emp;
LOOP
FETCH c_emp INTO v_id, v_name, v_sal;
EXIT WHEN c_emp%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(v_id || ' ' || v_name || ' ' || v_sal);
END LOOP;
CLOSE c_emp;
END;
/
```

2. Implicit cursor (count rows):
```
DECLARE
v_count NUMBER;
BEGIN
SELECT COUNT(*) INTO v_count FROM Employee;
DBMS_OUTPUT.PUT_LINE('Total rows = ' || v_count);
END;
/
```

3. Parameterized cursor:
```
DECLARE
v_empid NUMBER := &amp;emp;_id;
CURSOR c_sal(p_id NUMBER) IS SELECT Salary FROM Employee WHERE Emp_id = p_id;
v_sal Employee.Salary%TYPE;
BEGIN
OPEN c_sal(v_empid);
```

```
FETCH c_sal INTO v_sal;
IF c_sal%FOUND THEN
DBMS_OUTPUT.PUT_LINE('Salary = ' || v_sal);
ELSE
DBMS_OUTPUT.PUT_LINE('Employee not found');
END IF;
CLOSE c_sal;
END;
/
```

Problem Statement 10 — CRUD (MongoDB Social_Media)

Collection: Social_Media(User_Id, User_Name, No_of_Posts, No_of_Friends, Friends_List, Interests)

```
db.createCollection("Social_Media");
-- insert sample documents (20 total recommended)
```

Queries:

```
1. db.Social_Media.find().forEach(doc => printjson(doc));
2. db.Social_Media.find({ No_of_Posts: { $gt: 100 } });
3. db.Social_Media.find({}, { User_Name:1, Friends_List:1, _id:0 });
4. db.Social_Media.find({ No_of_Friends: { $gt: 5 } }, { User_Id:1, Friends_List:1, _id:0 });
5. db.Social_Media.find().sort({ No_of_Posts: -1 });
```

Problem Statement 11 — DDL using MySQL (repeat of PS2 style)

Tasks:

```
1. Create tables with referential integrity (see PS2).
2. ER diagram (Customer - Account - Branch).
3. CREATE INDEX idx_account_no ON Account(Account_no);
4. CREATE VIEW Customer_Info AS SELECT * FROM Customer WHERE Age < 45;
5. UPDATE Account SET date_open = '2017-04-16' WHERE Account_no = ;
6. CREATE SEQUENCE branch_seq START WITH 4 INCREMENT BY 1 NOCACHE;
7. CREATE SYNONYM Branch_info FOR Branch;
```

Problem Statement 12 — Triggers (PL/SQL)

Schema: Employee(emp_id, emp_name, DoJ, salary, commission, job_title)

1. Trigger to prevent salary decrease:

```
CREATE OR REPLACE TRIGGER trg_no_salary_decrease
BEFORE UPDATE OF salary ON Employee
FOR EACH ROW
BEGIN
IF :NEW.salary < :OLD.salary THEN
RAISE_APPLICATION_ERROR(-20001,'Salary cannot be decreased');
END IF;
END;
/
```

2. Trigger to log job title changes into job_history:

```
CREATE TABLE job_history ( emp_id NUMBER, old_job_title VARCHAR2(100), old_dept_id NUMBER, start_date
DATE, end_date DATE );
CREATE OR REPLACE TRIGGER trg_job_change
AFTER UPDATE OF job_title ON Employee
FOR EACH ROW
WHEN (OLD.job_title IS NOT NULL AND :OLD.job_title <> :NEW.job_title)
BEGIN
INSERT INTO job_history(emp_id, old_job_title, old_dept_id, start_date, end_date)
VALUES (:OLD.emp_id, :OLD.job_title, :OLD.dept_id, :OLD.DoJ, SYSDATE);
END;
```

/

Problem Statement 13 — MapReduce (MongoDB Student)

Collection: Student(roll_no, name, class, dept, aggregate_marks)

db.createCollection("Student");
db.Student.insertMany([...]);

Queries (aggregation):
1. db.Student.aggregate([{ $match: { class: "TE" } }, { $group: { _id: "$dept", TotalMarks: { $sum: "$aggregate_marks" } } }]);
2. db.Student.aggregate([{ $match: { class: "SE" } }, { $group: { _id: "$dept", HighestMarks: { $max: "$aggregate_marks" }, Student: { $first: "$name" } } }]);
3. db.Student.aggregate([{ $match: { class: "BE" } }, { $group: { _id: "$dept", AvgMarks: { $avg: "$aggregate_marks" } } }]);

Problem Statement 14 — DML using MySQL

Tasks:
1. ALTER TABLE Customer MODIFY Email_Address VARCHAR(20);
2. ALTER TABLE Customer MODIFY Email_Address VARCHAR(20) NOT NULL;
3. SELECT COUNT(DISTINCT c.CustID) AS total_customers FROM Customer c JOIN Account a ON c.CustID = a.CustID WHERE a.Balance > 50000;
4. SELECT AVG(Balance) AS avg_saving_balance FROM Account WHERE Account_type = 'Saving Account';
5. SELECT * FROM Customer WHERE Cust_Address = 'Pune' OR Name LIKE 'A%';
6. CREATE TABLE Saving_Account AS SELECT Account_no, BranchID, CustID, date_open, Balance FROM Account WHERE Account_type = 'Saving';
7. SELECT c.CustID, c.Name, c.Age, a.Balance FROM Customer c JOIN Account a ON c.CustID = a.CustID WHERE a.Balance >= 20000 ORDER BY c.Age;